下载APP

模型实战准备（二）| 模型特征、训练样本的处理

2020-11-16 王喆

深度学习推荐系统实战 进入课程 >

讲述：王喆
时长 14:12 大小 13.01M

你好，我是王喆，欢迎来到模型实战准备第二课。

这节课，我们来讲实战中所需的模型特征和训练样本的处理。为什么我们要专门花一节课的时间讲这些呢？因为在推荐模型篇中，我们的重点在于学习模型结构的原理和实现，而要实现并且训练这些模型，我们就必须先解决训练所需的样本和特征的处理的问题。

这节课我们先来把模型实战的准备工作做完。具体来说，今天，我会带你用 Spark 来处理深度学习模型训练所需的样本和特征，再把特征存储到 Redis 中，供模型线上服务时调用。

为什么选择 Spark 为 TensorFlow 处理特征和样本?

这个时候，你可能会有疑问，我们的深度学习模型将在 TensorFlow 上进行训练，为什么要用 Spark 处理样本？可不可以直接让 TensorFlow 解决所有的事情呢？这是一个好问题，在我们学习具体的技术之前，先解决这个架构上的疑问是很有必要的。

在业界的实践中，我们需要记住一个原则，就是**让合适的平台做合适的事情**。比如说，数据处理是 Spark 的专长，流处理是 Flink 的专长，构建和训练模型是 TensorFlow 的专长。在使用这些平台的时候，我们最好能够用其所长，避其所短。这也是一个业界应用拥有那么多个模块、平台的原因。

你可能想说，TensorFlow 也可以处理数据啊。没错，但是它并不擅长分布式的并行数据处理，在并行数据处理能力上，TensorFlow 很难和动辄拥有几百上千个节点的 Spark 相比。那在面对海量数据的时候，如果我们能够利用 Spark 进行数据清洗、数据预处理、特征提取的话，最好的方案就是让 Spark 发挥它的长处，承担“繁重”但相对简单的样本和特征处理的工作，为 TensorFlow 减轻负担。

物品和用户特征都有哪些?

既然我们决定用 Spark 进行样本和特征处理，那下一个问题又接踵而来，我们能从 MovieLens 的数据集中抽取出什么特征呢？这就要用到我们在特征工程篇中学到的，关于推荐系统特征以及相关特征处理方法的知识，如果你记得还不够扎实，我建议你可以先回去复习一下。

MovieLens 数据集中，可供我们提取特征的数据表有两个，分别是 movies 表和 ratings 表，它们的数据格式如下：

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy
6	Heat (1995)	Action|Crime|Thriller
7	Sabrina (1995)	Comedy|Romance
8	Tom and Huck (1995)	Adventure|Children
9	Sudden Death (1995)	Action
10	GoldenEye (1995)	Action|Adventure|Thriller

电影基本数据movies表

userId	movieId	rating	timestamp
1	2	3.5	1112486027
1	29	3.5	1112484676
1	32	3.5	1112484819
1	47	3.5	1112484727
1	50	3.5	1112484580
1	112	3.5	1094785740
1	151	4.0	1094785734
1	223	4.0	1112485573
1	253	4.0	1112484940
1	260	4.0	1112484826

用户评分数据ratings表

图1 电影基本数据movies表（左），用户评分数据ratings表（右）

接下来，我们按照“物品特征”“用户特征”“场景特征”，这三大类推荐系统特征的顺序，来看一看从这两张表中能提取出什么样的特征。

“物品特征”在我们的项目里指的就是电影特征了，从 movies 表中我们可以提取出电影的基本信息，包括 movieId、title（电影名）、releaseYear（发布年份）和 genre（风格类型）。除此之外，我们还可以利用 ratings 表为每个电影提取出一些统计类的特征，包括电影的平均评分、评分标准差等等。

接下来是“用户特征”，乍一看，从 movies 和 ratings 表中，除了 userId，我们好像找不到其他直接可用的用户信息了。这个时候，千万不要忘了我们之前提到过的，用户特征最重要的部分就是历史行为特征，所以，从用户的评分历史中，我们其实可以提取出非常多有价值的特征。比如，我们可以根据 ratings 表的历史联合 movies 表的电影信息，提取出用户统计类特征，它包括用户评分总数、用户平均评分、用户评分标准差、用户好评电影的发布年份均值、用户好评电影的发布年份标准差、用户最喜欢的电影风格，以及用户好评电影 ID 等等。

最后是“场景特征”，我们可用的场景特征就一个，那就是评分的时间戳，我们把它作为代表时间场景的特征放到特征工程中。

好了，到这儿，我们就梳理完了所有可用的特征，我把它们总结在了下面的表格里，供你参考。

特征类别	特征	特征ID
物品特征	电影ID	movieId
	电影发布年份	releaseYear
	电影风格类型	movieGenre
	电影评分总数	movieRatingCount
	电影平均评分	movieAvgRating
	电影评分标准差	movieRatingStddev
用户特征	用户ID	userId
	用户评分总数	userRatingCount
	用户平均评分	userAvgRating
	用户评分标准差	userRatingStddev
	用户好评电影的发布年份均值	userAvgReleaseYear
	用户好评电影的发布年份标准差	userReleaseYearStddev
	用户最喜欢的电影风格	userGenre
	用户好评电影ID	userRatedMovie
场景特征	评价时间戳	timestamp

图2 所有可用的特征汇总表

用 Spark 来提取这些特征的总体实现会比较琐碎，所以我就不把全部代码贴在这里了，你可以参考 SparrowRecsys 项目中的 com.wzhe.sparrowrecsys.offline.spark.featureeng.FeatureEngForRecModel 对象，里面包含了所有特征工程的代码。这里，我们只讲几个有代表性的统计型特征的处理方法。

复制代码

```
val movieRatingFeatures = samplesWithMovies3.groupBy(col("movieId"))
  .agg(count(lit(1)).as("movieRatingCount"),
   avg(col("rating")).as("movieAvgRating"),
   stddev(col("rating")).as("movieRatingStddev"))
```

计算统计型特征的典型方法，就是利用 Spark 中的 groupBy 操作，将原始评分数据按照 movieId 分组，然后用 agg 聚合操作来计算一些统计型特征。比如，在上面的代码中，我们就分别使用了 count 内置聚合函数来统计电影评价次数（movieRatingCount），用 avg 函数来统计评分均值（movieAvgRating），以及使用 stddev 函数来计算评价分数的标准差（movieRatingStddev）。

特征处理具体过程，我们就讲完了。不过，这里我还想和你多分享一些我的经验。**一般来说，我们不会人为预设哪个特征有用，哪个特征无用，而是让模型自己去判断，如果一个特征的加入没有提升模型效果，我们再去除这个特征。就像我刚才虽然提取了不少特征，但并不是说每个模型都会使用全部的特征，而是根据模型结构、模型效果有针对性地部分使用它们。**在接下来的课程中，我们还会详细探讨不同模型对这些特征的具体使用方法。

最终的训练样本是什么样的？

特征提取之后就到了训练模型的步骤，为了训练模型，我们还需要生成模型所需的训练样本。这里我们需要明确两件事情，一是样本从哪里来，二是样本的标签是什么。这两件事情都跟我们训练模型的目标有关系。

对于一个推荐模型来说，它的根本任务是预测一个用户 U 对一个物品 I 在场景 C 下的喜好分数。所以在训练时，我们要为模型生成一组包含 U、I、C 的特征，以及最终真实得分的样本。在 SparrowRecsys 中，这样的样本就是基于评分数据 ratings，联合用户、物品特征得来的。

其中，用户特征和物品特征都需要我们提前生成好，然后让它们与 ratings 数据进行 join 后，生成最终的训练样本，具体的实现也在 FeatureEngForRecModel 中，你可以先参考我在下面贴出的关键代码。这样，我们就解决了第一个关键问题。

```
//读取原始ratings数据
val ratingSamples = spark.read.format("csv").option("header", "true").load(rat
//添加样本标签
val ratingSamplesWithLabel = addSampleLabel(ratingSamples)
//添加物品（电影）特征
val samplesWithMovieFeatures = addMovieFeatures(movieSamples, ratingSamplesWit
//添加用户特征
val samplesWithUserFeatures = addUserFeatures(samplesWithMovieFeatures)
```

复制代码

接着，我们来看第二个关键问题，也就是样本的标签是什么，对于 MovieLens 数据集来说，用户对电影的评分是最直接的标签数据，因为它就是我们想要预测的用户对电影的评价，所以 ratings 表中的 0-5 的评分数据自然可以作为样本的标签。

但对于很多应用来说，我们基本上不可能拿到它们的评分数据，更多的是点击、观看、购买这些隐性的反馈数据，所以业界更多使用 CTR 预估这类解决二分类问题的模型去解决推荐问题。

为了让我们的实践过程更接近真实的应用场景，我也对 MovieLens 数据集进行了进一步处理。具体来说就是，把评分大于等于 3.5 分的样本标签标识为 1，意为“喜欢”，评分小于 3.5 分的样本标签标识为 0，意为“不喜欢”。这样一来，我们可以完全把推荐问题转换为 CTR 预估问题。

如何在生成样本时避免引入“未来信息”？

训练模型所需要的训练样本我们已经得到了，但是，在处理训练样本的时候，还有一个问题我们一定要注意，那就是引入未来信息（Future Information）的问题，这也是我们在实际工作中经常会遇到的问题。

什么叫做未来信息呢？如果我们在 t 时刻进行模型预测，那么 t+1 时刻的信息就是未来信息。这个问题在模型线上服务的时候是不存在的，因为未来的事情还未发生，我们不可能知道。但在离线训练的时候，我们就容易犯这样的错误。比如说，我们利用 t 时刻的样本

进行训练，但是使用了全量的样本生成特征，那么这些特征就包含了 t+1 时刻的未来信息，这就是一个典型的引入未来信息的错误例子。

这样说可能还是有点抽象，我们用刚才讲过的特征举个例子。刚才我们说到有一个用户特征叫做用户平均评分（userAvgRating），我们通过把用户评论过的电影评分取均值得到它。假设，一个用户今年评论过三部电影，分别是 11 月 1 日评价电影 A，评分为 3 分，11 月 2 日评价电影 B，评分为 4 分，11 月 3 日评价电影 C，评分为 5 分。如果让你利用电影 B 这条评价记录生成样本，样本中 userAvgRating 这个特征的值应该取多少呢？

有的同学会说，应该取评价过的电影评分的均值啊，（3+4+5）/3=4 分，应该取 4 分啊。这就错了，因为在样本 B 发生的时候，样本 C 还未产生啊，它属于未来信息，你怎么能把 C 的评分也加进去计算呢？而且样本 B 的评分也不应该加进去，因为 userAvgRating 指的是历史评分均值，B 的评分是我们要预估的值，也不可以加到历史评分中去，所以正确答案是 3 分，我们只能考虑电影 A 的评分。

因此，在处理历史行为相关的特征的时候，我们一定要考虑未来信息问题。类似的还有用户评分总数、用户评分标准差、用户最喜欢的电影风格、用户好评电影 ID 等一系列特征。

那在 Spark 中，我们应该如何处理这些跟历史行为相关的特征呢？这就需要用到 window 函数了。比如说，我在生成 userAvgRating 这个特征的时候，是使用下面的代码生成的：

复制代码

```
1 withColumn("userAvgRating", avg(col("rating"))
2   .over(Window.partitionBy("userId")
3     .orderBy(col("timestamp")).rowsBetween(-100, -1)))
4
```

我们可以看到，代码中有一个 over(Window.partitionBy(“userId”).orderBy(col(“timestamp”))) 操作，它的意思是，在做 rating 平均这个操作的时候，我们不要对这个 userId 下面的所有评分取平均值，而是要创建一个滑动窗口，先把这个用户下面的评分按照时间排序，再让这个滑动窗口一一滑动，滑动窗口的位置始终在当前 rating 前一个 rating 的位置。这样，我们再对滑动窗口内的分数做平均，就不会引入未来信息了。

类似的操作，我使用在了所有与历史行为有关的特征中，你也可以在 SparrowRecsys 的源码中看到。

如何把特征数据存入线上供模型服务用？

在生成好特征和训练样本之后，还有一个问题需要我们解决，那就是特征的线上存储问题。因为训练样本是供离线训练使用的，而线上模型推断过程是要使用线上特征的。

好在，特征数据库 Redis 已经为我们提供了解决办法。我们把用户特征和物品特征分别存入 Redis，线上推断的时候，再把所需的用户特征和物品特征分别取出，拼接成模型所需的特征向量就可以了。

FeatureEngForRecModel 中的 extractAndSaveUserFeaturesToRedis 函数给出了详细的 Redis 操作，我把其中的关键操作放在了下面。

复制代码

```
val userKey = userFeaturePrefix + sample.getAs[String]("userId")
val valueMap = mutable.Map[String, String]()
valueMap("userRatedMovie1") = sample.getAs[String]("userRatedMovie1")
valueMap("userRatedMovie2") = sample.getAs[String]("userRatedMovie2")
...
valueMap("userAvgRating") = sample.getAs[String]("userAvgRating")
valueMap("userRatingStddev") = sample.getAs[String]("userRatingStddev")


redisClient.hset(userKey, JavaConversions.mapAsJavaMap(valueMap))
```

我们可以看到，代码中使用了 Redis 一个新的操作 hset，它的作用是把一个 Map 存入 Redis。这样做有什么好处呢？对于这里的用户特征来说，Map 中存储的就是特征的键值对，又因为这个 Map 本身是 userId 的值，所以，每个 userId 都拥有一组用户特征。这样一来，我们就可以在推荐服务器内部，通过 userId 来取出所有对应的用户特征了。当然，物品特征的储存方式是一样的。

到这里，我们完成了所有特征工程相关的准备工作，为之后的模型训练也做好了充足的准备。

小结

这节课，我们选择 Spark 作为特征和样本处理的平台，是因为 Spark 更擅长海量数据的分布式处理，为 TensorFlow 减轻数据处理的负担。在选择具体特征的过程中，我们遵循了“物品特征”“用户特征”“场景特征”这三大类特征分类方式，基于 MovieLens 的 ratings 表和 movies 表完成了特征抽取。

在样本处理过程中，我们选用评分和基于评分生成的好评差评标识作为样本标签，并基于 ratings 表的每条数据，通过联合物品和用户数据生成训练样本。在训练样本的生成中，要特别注意“未来信息”的问题，利用 Spark 中的 window 函数滑动生成历史行为相关特征。最后我们利用 Redis 的 hset 操作把线上推断用的特征存储 Redis。

这些重点内容，我也都总结在了下面的表格里，你可以看一看。

知识点	关键描述
选用Spark作为 特征处理平台的原因	相比TensorFlow，Spark更擅长海量数据的分布式处理， 可以为TensorFlow减轻数据处理的负担
物品和用户特征	物品特征：电影ID、电影发布年份、电影风格类型、电影评分总数、电影平均评分、电影评分标准差 用户特征：用户ID、用户评分总数、用户平均评分、用户评分标准差、用户好评电影的发布年份均值、用户好评电影的发布年份标准差、用户最喜欢的电影风格、用户好评电影ID 场景特征：评价时间戳
训练样本的标签生成	Ratings表本身的评分值， 或基于评分值生成的好评、差评标签
未来信息的概念	在t时刻进行模型预测，那么t+1时刻的信息就是未来信息
在Spark中避免引入未来信息的方式	使用window函数处理样本特征
把特征存入向上数据库的方式	使用Redis的hset操作完成用户和物品特征存储

极客时间

课后思考

为了避免引入未来信息，咱们课程中讲了基于 userId 的 window 函数方法，你觉得还有哪些方法也能避免引入未来信息吗？

欢迎把你的思考和答案写在留言区，如果你觉得今天的内容对你有所帮助，也欢迎你分享给你的朋友或同事。我们下节课见！

提建议

更多学习推荐

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 模型实战准备（一）| TensorFlow入门和环境配置

下一篇 17 | Embedding+MLP：如何用TensorFlow实现经典的深度学习模型？

精选留言 (7)

写留言

onepencil
2020-11-16

线上会组装当前访问时的历史序列及历史平均值等，可以利用flink实时落盘这些线上特征，这样线下就不用再离线生成，也就杜绝了特征穿越问题

展开

作者回复: 赞，这是业界的解决方法之一，存储特征的snapshot。

8

Sebastian
2020-11-16

老师好，想问下使用hset把用户和物品的特征存入redis的场景下，如果用户上亿物品也上亿的话，存入redis会使用很大的资源。实际场景下并不是所有用户都是活跃用户，很多情况下可能用不上。那是否是在redis只存入活跃用户的特征呢？但是如果这样解决，遇到非活跃用户是否就没有特征值了？

...

展开

作者回复: 我在存储模块那节课说过，SparrowRecsys项目确实对存储模块进行了一定程度的简化，实际应用中还是要多考虑分级存储，redis实际上当作一个缓存层来使用。

关于大量数据key value数据的存储和线上查找，可以多调研rocksdb，cassandra，dynamodb和mongodb。

1

萬里長空
2020-11-20

老师，问一个spark-udf的问题，看官网上定义udf的时候要使用register，但是您这里并没有使用，是sql与DataFrame的区别吗？

展开

作者回复: 是的，是spark sql和直接操作dataframe的区别。直接操作dataframe不用register

Wiiki
2020-11-19

王老师，我按照您的示例代码，将用户特征存入redis(调用extractAndSaveUserFeaturesToRedis(samplesWithUserFeatures)的时候报错了，首先是报错：java.lang.Integer cannot be cast to java.lang.String，我在调试的过程中发现valueMap("userAvgReleaseYear") = sample.getAs[String]("userAvgReleaseYear")这个字段并没有将年份转换成String，还是原来的Int类型，所以改成 valueMap("userAvgReleaseYear") = sample.get...

展开

作者回复: 第一个问题确实是个bug，我已经提交了修改。
第二个问题我这边不存在，可能还需要你来debug一下。确认本地的redis环境都配置好了吧？

2

fsc2016
2020-11-18

老师，有俩个问题请教您：
1，前面章节生成了用户和物品的embding，这些embding也会作为一个特征字段加入到本节生成的用户和物品特征中嘛
2，本节提取了用户和物品特征，那进行ctr预估的话，是根据rateing表中，为每条记录添加对应提取出的用户特征和物品特征，作为一条训练记录。这样组合的话，是不是使用...
展开

作者回复: 1. 可以的，也是常用的做法，就是预训练embedding+深度模型的做法

2. 这个问题不是特别理解，我已经在这节专门讲了如何用window函数生成用户和物品特征，来消除未来信息，应该是回答了你这个问题?

1

Jay
2020-11-18

刚入坑，好像一直没有看到关于特征组合的内容，这个点一直没有理解清楚，啥时候会讲讲呢?

作者回复: 组合特征当然可以通过人工组合的手段进行组合，这点我们仍然可以沿用之前传统机器学习的经验。

对于深度学习模型来说，也可以交由模型的内部结构来组合，这一点后面关于模型的讲解还会涉及到。

那时刻
2020-11-17

把评分大于等于 3.5 分的样本标签标识为 1，意为“喜欢”，评分小于 3.5 分的样本标签标识为 0，意为“不喜欢”。这样可以完全把推荐问题转换为 CTR 预估问题。请问老师，3.5分这个值是怎么来的呢?

另外一个问题，我们把评分时间戳作为代表时间场景的特征放到特征工程中。请问把时...

展开

作者回复: 两个问题都很好，希望大家都有这样实践中的思考。

第一个问题。基本原则是我在分析完分数的总体分布后得出的，3.5分基本是正负样本比例1:1的分界线，另外大于3.5分也符合我们直观意义上的高分，所以认为3.5分是比较合理的。

第二个问题。确实像你说的这样，我们实践中经常从时间戳中提取出周末，假日，季节这些特征，因为这些时间特征往往影响着数据的一些潜在模式，所以还是非常有价值的。

2